Ex-7 Perfomance analysis on KNN classification technique

October 26, 2023

URK21CS1128

AIM: To demonstrate performance analysis on KNN classification technique

DESCRIPTION: K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well

K-nearest neighbors (KNN) algorithm uses feature similarity to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. We can understand its working with the help of following steps:

Step1: For implementing any algorithm, we need dataset. So, during the first step of KNN, load the training as well as test data. Step2: Choose the value of K i.e. the nearest data points. K can be any integer. Step3: For each point in the test data do the following:

3.1: Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean. 3.2: Now, based on the distance value, sort them in ascending order. 3.3: Next, it will choose the top K rows from the sorted array. 3.4: Now, it will assign a class to the test point based on most frequent class of these rows. Step4: End Specificity, in contrast to recall, may be defined as the number of negatives returned by the classification model.

Support may be defined as the number of samples of the true response that lies in each class of target values. F1 Score This score will give us the harmonic mean of precision and recall.

Mathematically, F1 score is the weighted average of the precision and recall. The best value of F1 would be 1 and worst would be 0. F1 score will be calculated with the help of following formula: F1 = 2 ∗ (precision ∗ recall) / (precision + recall) F1 score is having equal relative contribution of precision and recall. classification_report function of sklearn.metrics is used to get the classification report of classification model. AUC (Area Under Curve)-ROC (Receiver Operating Characteristic) is a performance metric, based on varying threshold values, for classification problems. As name suggests, ROC is a prob- ability curve and AUC measure the separability. In simple words, AUC-ROC metric will tell us about the capability of model in distinguishing the classes. Higher the AUC, better the model. Mathematically, it can be created by plotting TPR (True Positive Rate) i.e. Sensitivity or recall vs FPR (False Positive Rate) i.e. 1-Specificity, at various threshold values. roc_auc_score function of sklearn.metrics is used to compute AUC-ROC.

[3]:
```
import pandas as pd
import numpy as np
```

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix,␣
↪accuracy_score,classification_report, roc_auc_score
```

1. Develop a KNN classification model for the wine dataset using the scikit-learn

a. Read the data

[4]:
```
print("URK21CS1128")
data = pd.read_csv('wine.csv')
data.head(2)
```

```
URK21CS1128
```

[4]:
```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70          0.0             1.9      0.076
1            7.8              0.88          0.0             2.6      0.098

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                    34   0.9978  3.51       0.56
1                 25.0                    67   0.9968  3.20       0.68

   alcohol quality
0      9.4     bad
1      9.8     bad
```

b. Data Cleaning
    a. Replace 0 in [‘chlorides’, ‘density’, ‘pH’, ‘sulphates’] column with NaNvalue
c. Identify the columns with null value
d. Filling the null values by imputing the mean values in the corresponding column

[7]:
```
cols_to_replace_zero = ['chlorides', 'density', 'pH', 'sulphates']
data[cols_to_replace_zero] = data[cols_to_replace_zero].replace(0, float('nan'))
columns_with_null = data.columns[data.isnull().any()].tolist()
data[columns_with_null] = data[columns_with_null].fillna(data.
↪mean(numeric_only=True))
data.head(10)
```

[7]:
```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076
5            7.4              0.66         0.00             1.8      0.075
6            7.9              0.60         0.06             1.6      0.069
7            7.3              0.65         0.00             1.2      0.065
8            7.8              0.58         0.02             2.0      0.073
```

```
9            7.5              0.50         0.36              6.1       0.071

   free sulfur dioxide  total sulfur dioxide  density        pH  sulphates  \
0                 11.0                    34   0.9978  3.510000       0.56
1                 25.0                    67   0.9968  3.200000       0.68
2                 15.0                    54   0.9970  3.260000       0.65
3                 17.0                    60   0.9980  3.160000       0.58
4                 11.0                    34   0.9978  3.510000       0.56
5                 13.0                    40   0.9978  3.510000       0.56
6                 15.0                    59   0.9964  3.299488       0.46
7                 15.0                    21   0.9946  3.390000       0.47
8                  9.0                    18   0.9968  3.360000       0.57
9                 17.0                   102   0.9978  3.350000       0.80

   alcohol quality
0      9.4     bad
1      9.8     bad
2      9.8     bad
3      9.8    good
4      9.4     bad
5      9.4     bad
6      9.4     bad
7     10.0    good
8      9.5    good
9     10.5     bad
```

c. Use the columns: [‘fixed acidity’, ‘volatile acidity’, ‘citric acid’, ‘residual sugar’, ‘chlorides’, ‘free sulfur dioxide’] as the independent variables

[8]:
```
print("URK21CS1128")
X = data[['fixed acidity', 'volatile acidity', 'citric acid', 'residual␣
 ↪sugar','chlorides', 'free sulfur dioxide']]
X.head(2)
```

```
URK21CS1128
```

[8]:
```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70          0.0             1.9      0.076
1            7.8              0.88          0.0             2.6      0.098

   free sulfur dioxide
0                 11.0
1                 25.0
```

d. Use the target variable as ‘quality’ (‘good’ and ‘bad’ based on score >5 and <5)

[9]:
```
print("URK21CS1128")
y = data['quality']
y.head(4)
```

```
URK21CS1128
```

[9]:
```
0     bad
1     bad
2     bad
3    good
Name: quality, dtype: object
```

e. Encode the categorical value of the target column to numerical value

[10]:
```
print("URK21CS1128")
y = y.replace({'good':0, 'bad':1})
y.head(4)
```

```
URK21CS1128
```

[10]:
```
0    1
1    1
2    1
3    0
Name: quality, dtype: int64
```

f. Divide the data into training (75%) and testing set (25%)

[11]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
↪25,random_state=42)
```

g. Perform the classification with K=3

[12]:
```
print("URK21CS1128")
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
```

```
URK21CS1128
```

[12]:
```
KNeighborsClassifier(n_neighbors=3)
```

h. Analyse the performance of the classifier with various performance measures␣and display such as confusion matrix, accuracy, recall, precision,␣specificity, f-score, Receiver operating characteristic (ROC) curve and Area␣Under Curve (AUC) score

[13]:
```
print("URK21CS1128")
y_pred = knn.predict(X_test)
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("AUC Score:", roc_auc_score(y_test, y_pred))
```

```
URK21CS1128
Confusion Matrix:
 [[73 61]
 [37 79]]
```

```
Accuracy: 0.608
Classification Report:
               precision    recall  f1-score   support

           0       0.66      0.54      0.60       134
           1       0.56      0.68      0.62       116

    accuracy                           0.61       250
   macro avg       0.61      0.61      0.61       250
weighted avg       0.62      0.61      0.61       250

AUC Score: 0.6129053010808028
```

i. Change the value of K in KNN with 5,7,9,11 and tabulate the various TP, TN,␣ accuracy, f-score and AUC score obtained

[14]:
```
print("URK21CS1128")
for i in [5,7,9,11]:
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("Classification Report:\n", classification_report(y_test, y_pred))
    print("AUC Score:", roc_auc_score(y_test, y_pred))
```

```
URK21CS1128
Confusion Matrix:
 [[68 66]
 [46 70]]
Accuracy: 0.552
Classification Report:
               precision    recall  f1-score   support

           0       0.60      0.51      0.55       134
           1       0.51      0.60      0.56       116

    accuracy                           0.55       250
   macro avg       0.56      0.56      0.55       250
weighted avg       0.56      0.55      0.55       250

AUC Score: 0.5554554812146166
Confusion Matrix:
 [[64 70]
 [43 73]]
Accuracy: 0.548
Classification Report:
               precision    recall  f1-score   support
           0       0.60      0.48      0.53       134
           1       0.51      0.63      0.56       116

    accuracy                           0.55       250
   macro avg       0.55      0.55      0.55       250
weighted avg       0.56      0.55      0.55       250

AUC Score: 0.5534611425630469
Confusion Matrix:
 [[68 66]
 [42 74]]
Accuracy: 0.568
Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.51      0.56       134
           1       0.53      0.64      0.58       116

    accuracy                           0.57       250
   macro avg       0.57      0.57      0.57       250
weighted avg       0.58      0.57      0.57       250

AUC Score: 0.5726968605249615
Confusion Matrix:
 [[73 61]
 [47 69]]
Accuracy: 0.568
Classification Report:
              precision    recall  f1-score   support

           0       0.61      0.54      0.57       134
           1       0.53      0.59      0.56       116

    accuracy                           0.57       250
   macro avg       0.57      0.57      0.57       250
weighted avg       0.57      0.57      0.57       250

AUC Score: 0.5698018528049409
```

0.1 j. Analyse and infer for which K value, the classification algorithm provides␣

better performance.

K = 5 provides the highest accuracy and highest AUC score. K = 7 provides the second-highest accuracy and the second-highest AUC score. K = 9 provides the third-highest accuracy and the third-highest AUC score. K = 11 provides the lowest accuracy and but the lowest AUC score. K = 5 provides performance for this classification task. Therefore, K = 5 is considered the preferred choice for this specific problem.Therefore, we can infer that K = 5 provides better performance for this particular dataset ,classification problem.

```
[ ]: Result : Thus the program has been executed and the output has been verified␣
       ↪successfully
```